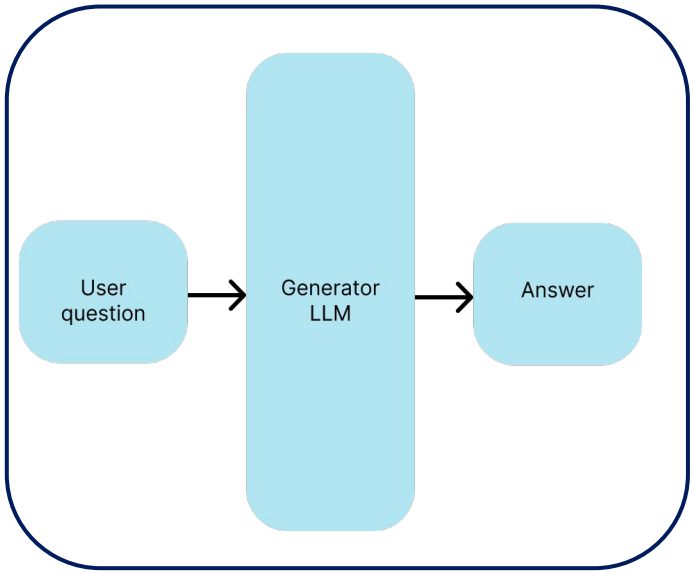
III. Retrieval Augmented Generation

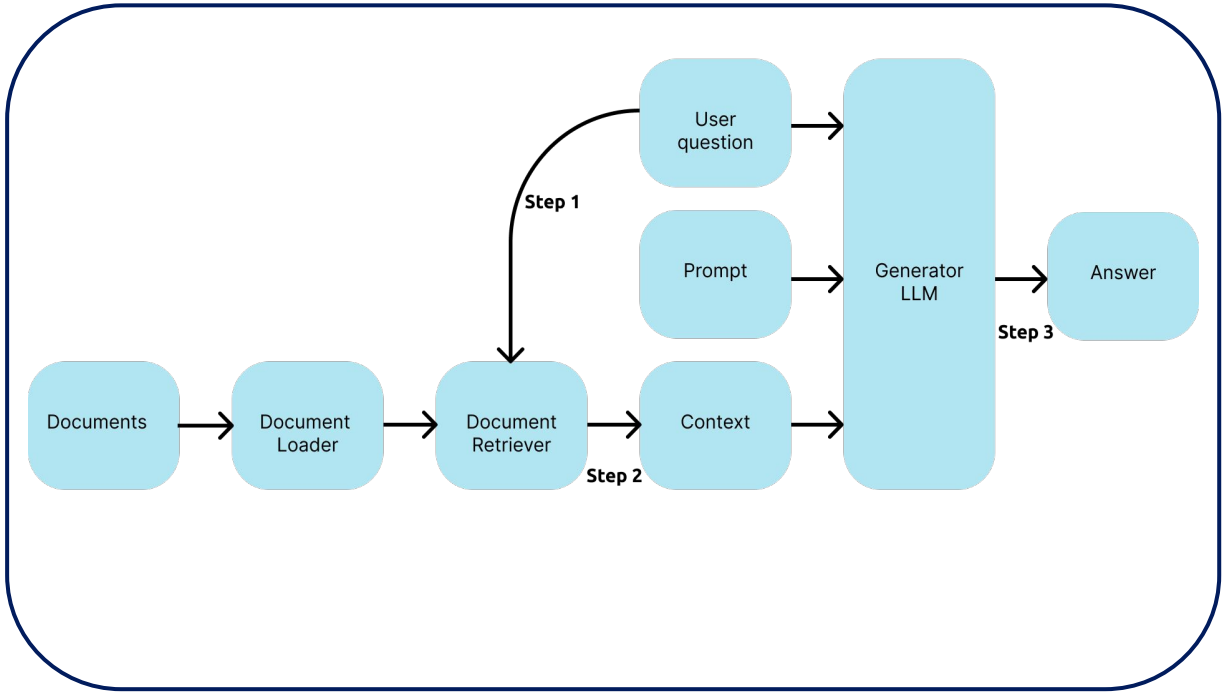
1. Basic Architecture
2. Information retrieval
3. Vectorstore & Search optimization
4. RAG Techniques
5. Evaluation
6. Multimodal RAG
7. SOTA RAG architectures

Dauphine | PSL
TUNIS

LLM vs RAG



LLM



RAG

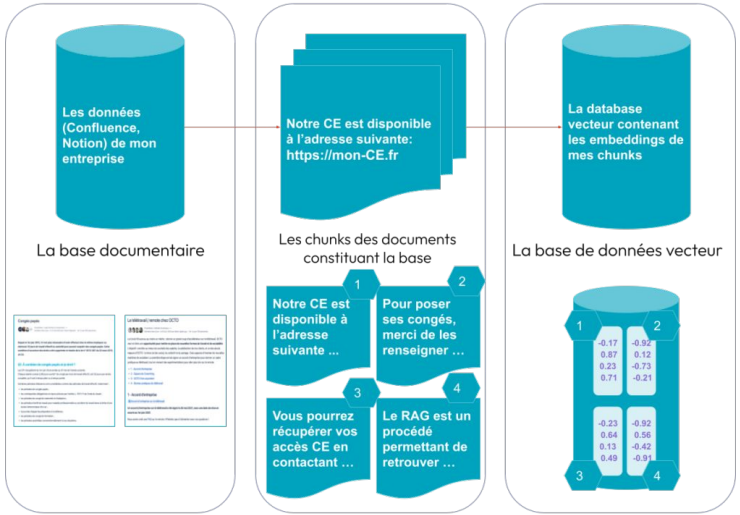
RAG Definition

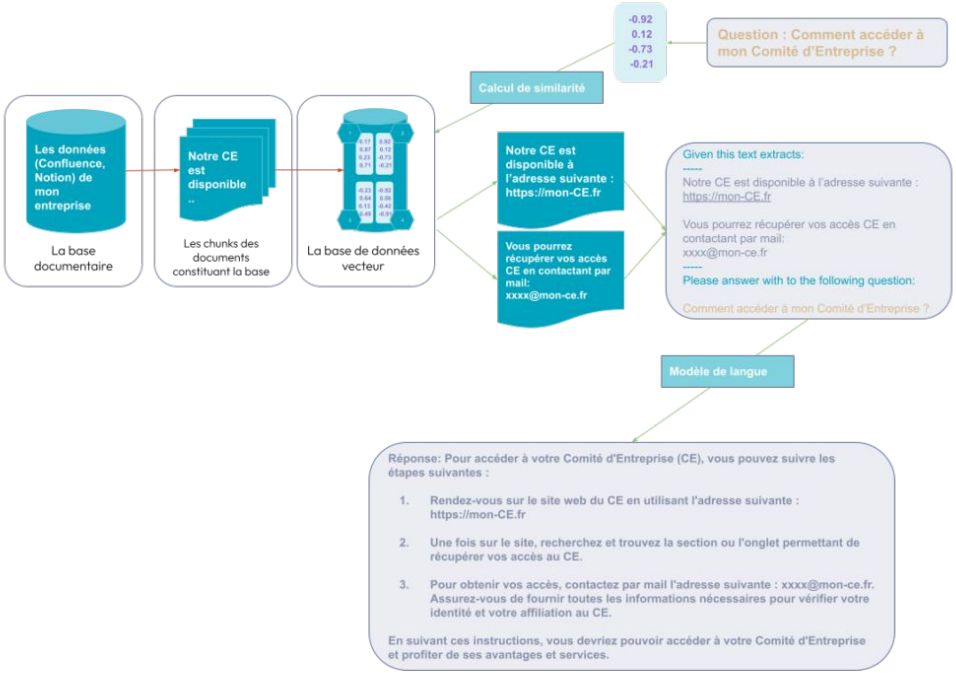
Definition: Retrieval-Augmented Generation (RAG) is a framework that combines retrieval-based and generation-based models. It enhances the capabilities of language models by providing them with access to external knowledge bases or documents during the generation process. This allows the model to generate more accurate and up-to-date information by retrieving relevant data instead of relying solely on its internal parameters.

Benefits:

- Produces more informed and factual responses.
- Can handle queries about recent events not present in the training data.
- Reduces hallucinations common in language models.

RAG Architecture



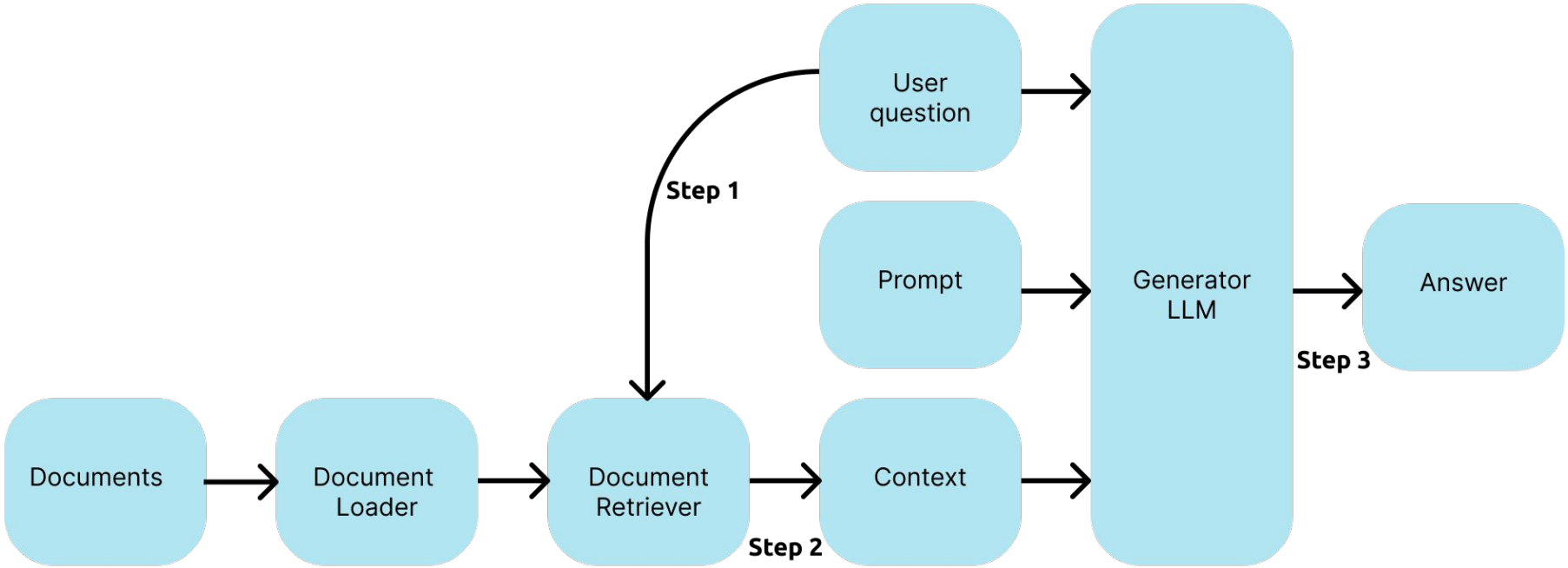


Step 1: Document ingestion

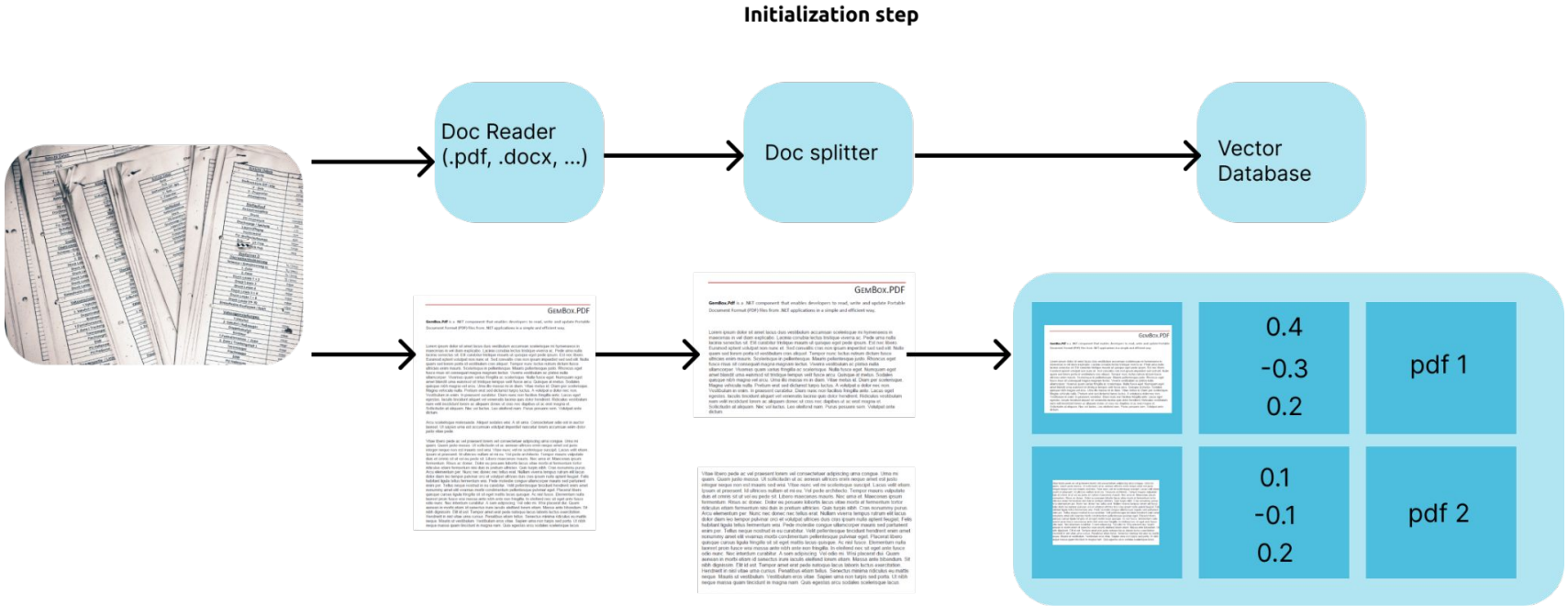
Step 2: Contextualized answering

[Construire son RAG (Retrieval Augmented Generation) grâce à langchain: L'exemple de l'Helpdesk d'OCTO]

RAG Architecture

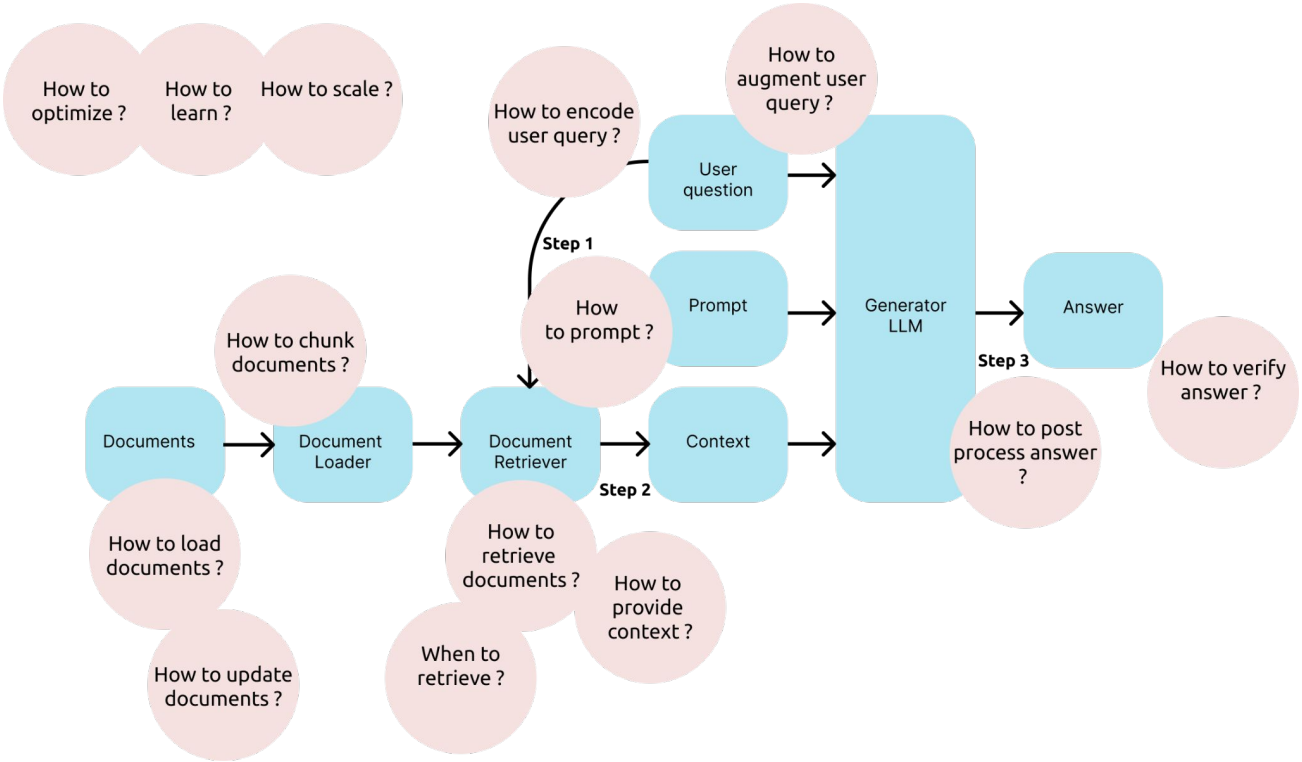


RAG Architecture



Dauphine | PSL TUNIS

How to ?



How to retrieve documents ?

TF-IDF

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

$\mathtt{tf}(t,d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

x

$\mathtt{idf}(t,D)$

blue	bright	can	see	shining	sky	sun	today
0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
- Most important word for each document is highlighted

$\mathtt{tfidf}(t,d,D) = \mathtt{tf}(t,d) \cdot \mathtt{idf}(t,D)$

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

TF-IDF score computation. [Image Source]

BM 25

Given a query Q, containing keywords {q_1, …, q_n}, the BM25 score of a document D is:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

- f(qi,D) is the number of times that the keyword qi occurs in the document *D*,
- |D| is the length of the document *D* in words
- avgdl is the average document length in the text collection from which documents are drawn.
- K_1 and *b* are free parameters, usually chosen, in absence of an advanced optimization, as $K_1 \in [1.2,2.0]$ and b=0.75
- *N* is the total number of documents in the collection, and
- n(qi) s the number of documents containing q_i

Text Search using TF-IDF and Elasticsearch

How to retrieve documents ?

Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Euclidean distance

$$d(p, q) = \|p - q\|.$$

How to retrieve documents ?

Maximal Marginal Relevance

$$MMR \stackrel{def}{=} Arg \max_{D_i \in R \setminus S} \left[\lambda (Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j)) \right]$$

The goal of this metric is to retrieve dissimilar documents and increase diversity

- D is the set of all candidate documents, R is the set of already selected documents, q is the query
- Sim_1 is the similarity function between a document and the query
- Sim_2 is the similarity function between two documents.
- di and dj are documents in D and R respectively

The parameter λ (mmr_threshold) controls the trade-off between relevance (the first term) and diversity (the second term). If mmr_threshold is close to 1, more emphasis is put on relevance, while a mmr_threshold close to 0 puts more emphasis on diversity.

The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries

How to retrieve documents ?

	big	count	create	dataset	differnt	features	hello	is	james	my	name	notebook	of	python	this	to	try	trying	vectorizer	words
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0
3	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	1
4	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0

Sparse vs Dense retrieval

Sparse embedding
(lots of 0)

Sparse Retrieval (TF IDF, BM 25, …) are methods to retrieve similar documents based on **keywords only.**
Dense Retrieval (Cos Sim, Euclidean distance) allows to retrieve document using semantic embedding representation of documents and query.

Hybrid Search is a method involving both sparse and dense retrievers to provide both advantages of the two approaches

Q. If I want to retrieve document based on the user query '*LeCun Meta*', what kind of retriever do I use ?
Q. If I want to retrieve document based on the user query '*What are the most wonderful shots of Lebron James ?*', what kind of retriever do I use ?
Q. If I want to retrieve document based on the user query '*What is the capital city of the biggest city in the world ?*', what kind of retriever do I use ?

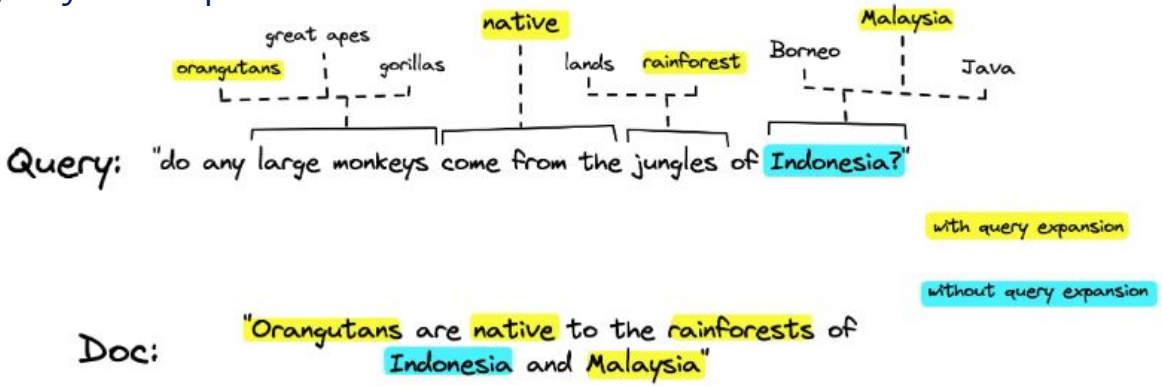
How to retrieve documents ?

Sparse Lexical and Expansion (SPLADE)

Vector database are super efficient compare to Splade at the moment
With sparse methods, you cannot get synonyms from a words.

SPLADE:

- Use Bert to get similar words like synonyms
- Provide these synonyms to a sparse methods



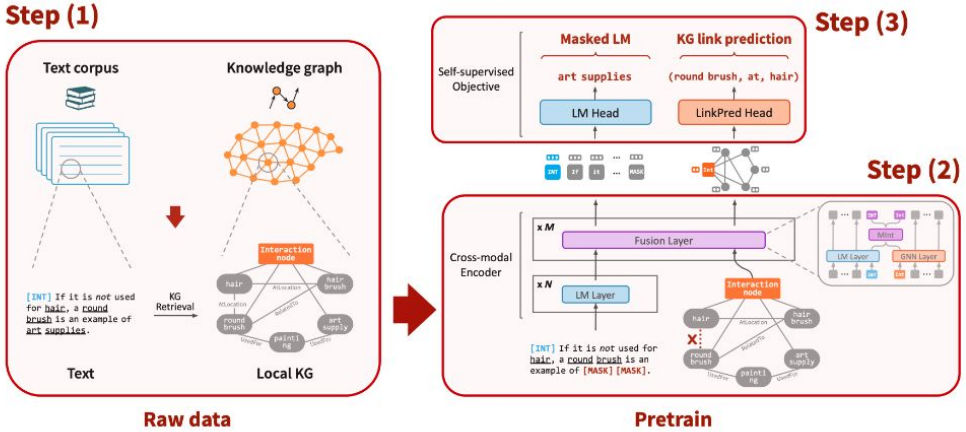
SPLADE for Sparse Vector Search Explained

Formal & AI, 2021, SPLADE V2

How to retrieve documents ?

Deep Bidirectional Language-Knowledge Graph Pretraining (DRAGON)

Dense retriever
Progressive Data Augmentation strategy for training sampling very difficult negatives



Lin & Al, 2023, How to Train Your DRAGON

Yasunaga, 2023, DRAGON: Training a Foundation Model from Text and Knowledge Graph [Blog]

How to retrieve documents ?

Best retrieval methods

Leaderboard for best Information Retrieval methods:
https://eval.ai/web/challenges/challenge-page/1897/leaderboard/4475

B - Baseline * - Private V - Verified Visible Metrics

Rank	Participant team	Avg (↑)	TREC-COVID (↑)	BioASQ (↑)	NFCorpus (↑)	NQ (↑)	HotpotQA (↑)	FiQA (↑)	Signal-1M (↑)	TREC-NEWS (↑)	Robust04 (↑)	ArguAna (↑)	Touche-2020 (↑)
1	nle ((BM25+SPLADE) RANKT5 top 50)	0.553	0.839	0.593	0.382	0.637	0.769	0.451	0.337	0.520	0.596	0.554	0.361
2	ZA+NM+Unicamp (InParsv2)	0.546	0.846	0.595	0.385	0.638	0.791	0.509	0.308	0.490	0.632	0.369	0.291
3	MetaAI+UW*2 (DRAGON+)	0.474	0.759	0.433	0.339	0.537	0.662	0.356	0.301	0.444	0.479	0.469	0.263
4	BEIR (SPLADE) V B	0.474	0.711	0.504	0.345	0.544	0.686	0.351	0.296	0.394	0.458	0.521	0.244
5	BEIR (BM25 multifield) B V	0.429	0.656	0.465	0.325	0.329	0.603	0.236	0.330	0.398	0.407	0.414	0.367

How to retrieve documents ?

Vector Database

Definition: A vector database is a specialized database designed to store, manage, and query high-dimensional vector embeddings of data such as text, images, or other content types.
These embeddings are numerical representations produced by machine learning models that capture the semantic meaning of the data.

A comparison of leading vector databases

	Pinecone	Weaviate	Milvus	Qdrant	Chroma	Elasticsearch	PGvector
Is open source	✗	✓	✓	✓	✓	✗	✓
Self-host	✗	✓	✓	✓	✓	✓	✓
Cloud management	✓	✓	✓	✓	✗	✓	(✓)
Purpose-built for Vectors	✓	✓	✓	✓	✓	✗	✗
Developer experience	👍👍👍	👍👍	👍👍	👍👍	👍👍	👍	👍
Community	Community page & events	8k☆ github, 4k slack	23k☆ github, 4k slack	13k☆ github, 3k discord	9k☆ github, 6k discord	23k slack	6k☆ github
Queries per second (using text nytimes-256-angular)	150 *for p2, but more pods can be added	791	2406	326	?	700-100 *from various reports	141
Latency, ms (Recall/Percentile 95 (millis), nytimes-256-angular)	1 *batched search, 0.99 recall, 200k SBERT	2	1	4	?	?	8
Supported index types	?	HNSW	Multiple (11 total)	HNSW	HNSW	HNSW	HNSW|IVFFlat
Hybrid Search (i.e. scalar filtering)	✓	✓	✓	✓	✓	✓	✓
Disk index support	✓	✓	✓	✓	✓	✗	✓
Role-based access control	✓	✗	✓	✗	✗	✓	✗
Dynamic segment placement vs. static data sharding	?	Static sharding	Dynamic segment placement	Static sharding	Dynamic segment placement	Static sharding	-
Free hosted tier	✓	✓	✓	(free self-hosted)	(free self-hosted)	(free self-hosted)	(varies)
Pricing (50k vectors @1536)	$70	fr. $25	fr. $65	est. $9	Varies	$95	Varies
Pricing (20M vectors, 20M req. @768)	$227 ($2074 for high performance)	$1536	fr. $309 ($2291 for high performance)	fr. $281 ($820 for high performance)	Varies	est. $1225	Varies

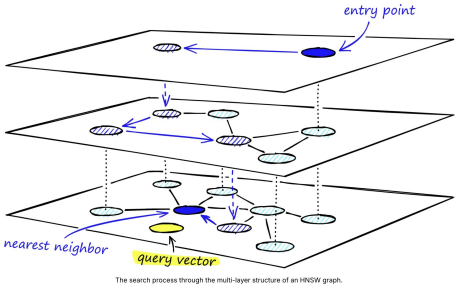
Vector DB Comparison

How to optimize ?

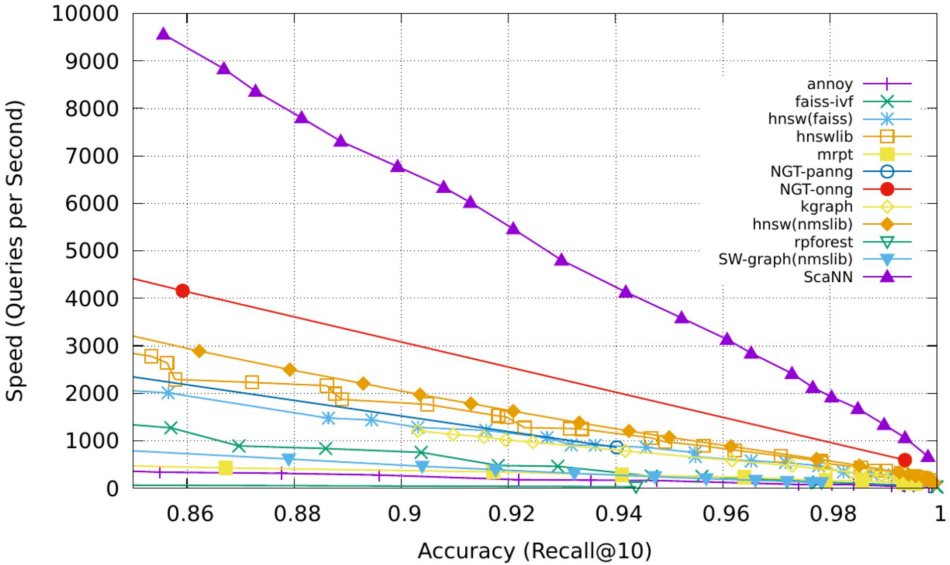
Efficient similarity search

- Scalable Nearest Neighbors (ScaNN) - Google
- Facebook AI Similarity Search (FAISS)
- Hierarchical Navigable Small Worlds (HNSW)

Definition:
ScaNN, FAISS and HNSW are methods for retrieving similar embeddings based on vector quantization and ANN search instead of full scan search.



Hierarchical Navigable Small Worlds (HNSW) [Blog]



Announcing ScaNN: Efficient Vector Similarity Search

How to provide context ?

Reciprocal Rank Fusion (RRF)

Definition:
RRF allows to merge results of different retrievers

$$\mathrm{RRF}(d) = \sum_{s=1}^{S} \frac{1}{k + r_s(d)}$$

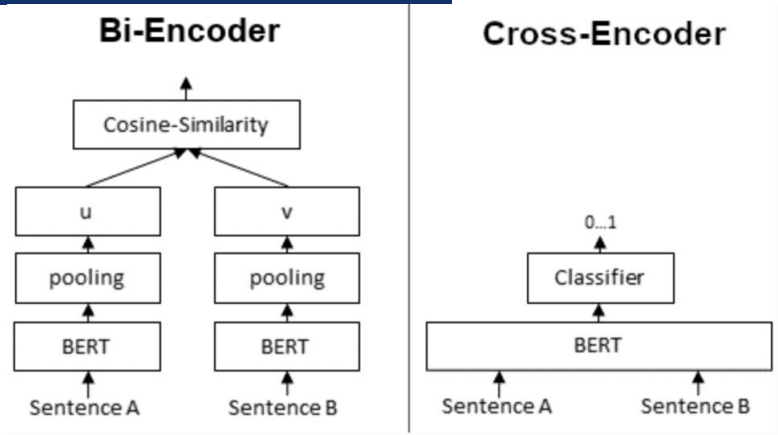
$\mathrm{RRF}(d)$: Denotes the RRF score for document d

$\sum_{s=1}^{S}$: Summation over all systems from s = 1 to S

$\frac{1}{k + r_s(d)}$: The reciprocal rank for document d in system s , adjusted by the constant k

How to provide context ?

Reranker



A reeranker rerank retrieved documents after a first similarity search

Reranker type:

- **Cross-Encoders**
- **Neural Rerankers**

Benefits of Using a Reranker:

- Increased Accuracy: Improves the likelihood that the most relevant information is used in generating the response.
- Better Contextual Understanding: Helps the system understand subtle nuances in the query.

Challenges:

- Computational Overhead: Additional processing can increase response time.
- Resource Intensive: Advanced models require significant computational resources.

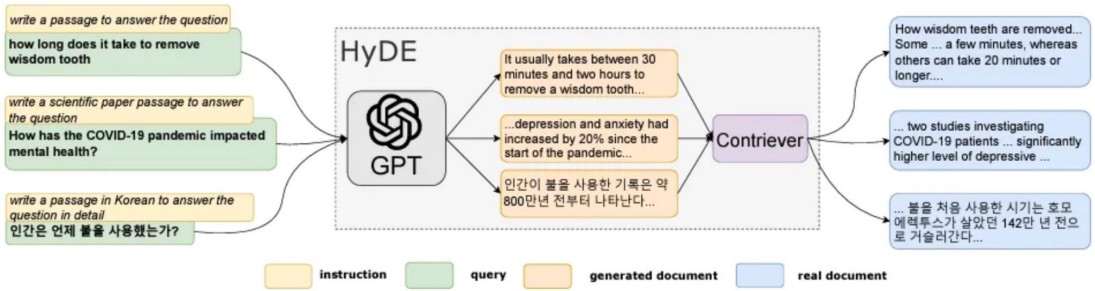
Bi Encoder vs Cross Encoder

How to augment user query ?

Query augmentation

Definition: query augmentation refers to the process of enhancing or expanding the user's original query to improve the retrieval of relevant documents or information from a knowledge base.
By augmenting the query, the system aims to retrieve more comprehensive and pertinent data, which can then be used to generate more accurate and informative responses.

HyDE: Generate a fake answer from a query to improve information retrieval

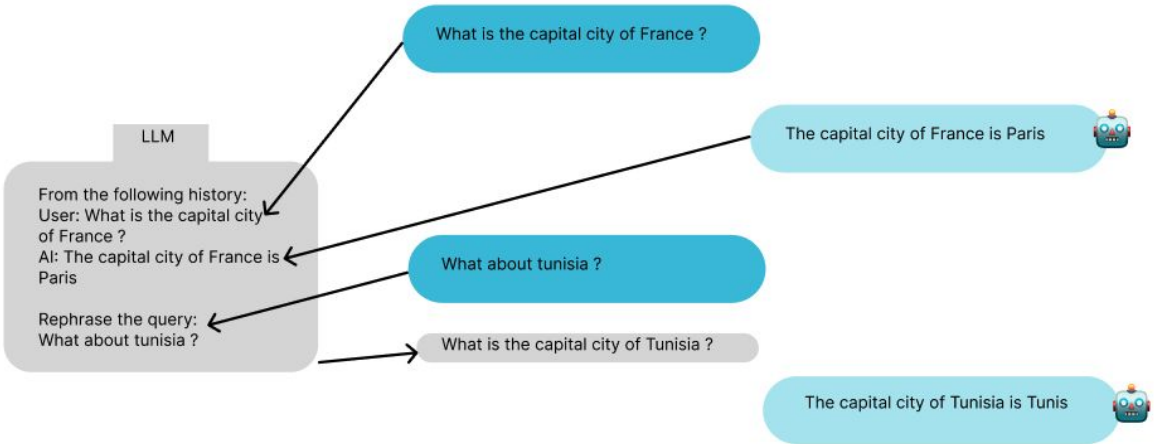


Luyu Gao & Al, 2022 Precise Zero-Shot Dense Retrieval without Relevance Labels

How to augment user query ?

Query rephrasing

Query rephrasing can be used to rephrase the query from the conversation history



How to provide context ?

Lost In the Middle

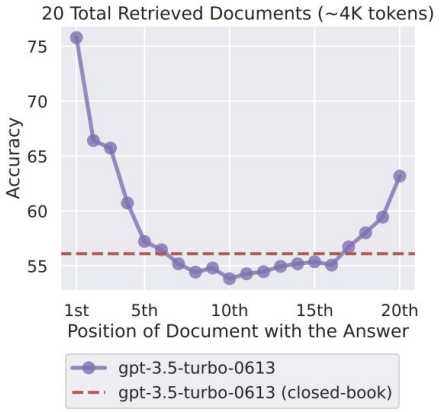
Retrieved context provided at the beginning or the end of the prompt have more impact on the answer

```python
from langchain.chains import LLMChain, StuffDocumentsChain
from langchain_chroma import Chroma
from langchain_community.document_transformers import (
    LongContextReorder,
)
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_core.prompts import PromptTemplate
from langchain_openai import OpenAI

# Get embeddings.
embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")

texts = [
    "Basquetball is a great sport.",
    "Fly me to the moon is one of my favourite songs.",
    "The Celtics are my favourite team.",
    "This is a document about the Boston Celtics",
    "I simply love going to the movies",
    "The Boston Celtics won the game by 20 points",
    "This is just a random text.",
    "Elden Ring is one of the best games in the last 15 years.",
    "L. Kornet is one of the best Celtics players.",
    "Larry Bird was an iconic NBA player.",
]

# Create a retriever
retriever = Chroma.from_texts(texts, embedding=embeddings).as_retriever(
    search_kwargs={"k": 10}
)
query = "What can you tell me about the Celtics?"

# Get relevant documents ordered by relevance score
docs = retriever.invoke(query)
docs
```

```
[Document(page_content='This is a document about the Boston Celtics'),
 Document(page_content='The Celtics are my favourite team.'),
 Document(page_content='L. Kornet is one of the best Celtics players.'),
 Document(page_content='The Boston Celtics won the game by 20 points'),
 Document(page_content='Larry Bird was an iconic NBA player.'),
 Document(page_content='Elden Ring is one of the best games in the last 15 years.'),
 Document(page_content='Basquetball is a great sport.'),
 Document(page_content='I simply love going to the movies'),
 Document(page_content='Fly me to the moon is one of my favourite songs.'),
 Document(page_content='This is just a random text.')]
```

```python
# Reorder the documents:
# Less relevant document will be at the middle of the list and more
# relevant elements at beginning / end.
reordering = LongContextReorder()
reordered_docs = reordering.transform_documents(docs)

# Confirm that the 4 relevant documents are at beginning and end.
reordered_docs
```

```
[Document(page_content='The Celtics are my favourite team.'),
 Document(page_content='The Boston Celtics won the game by 20 points'),
 Document(page_content='Elden Ring is one of the best games in the last 15 years.'),
 Document(page_content='I simply love going to the movies'),
 Document(page_content='This is just a random text.'),
 Document(page_content='Fly me to the moon is one of my favourite songs.'),
 Document(page_content='Basquetball is a great sport.'),
 Document(page_content='Larry Bird was an iconic NBA player.'),
 Document(page_content='L. Kornet is one of the best Celtics players.'),
 Document(page_content='This is a document about the Boston Celtics')]
```



LangChain, Long Context Reorder [Blog]

F. Liu & Al, 2023, Lost in the Middle: How Language Models Use Long Contexts

How to prompt ?

Prompt Engineering

- Write clear instructions
- Provide reference text
- Split complex tasks into simpler subtasks
- Give the model time to "think"
- Use external tools (RAG)

Tactic:

- Ask the model to adopt a persona
- Use delimiters to clearly indicate distinct parts of the input
- Specify the steps required to complete a task
- Provide examples
- Specify the desired length of the output

Open AI, Prompt engineering

How to load documents ?

Document Loader

Load any type of document (PDF, PPT(x), DOC(x), XLS(x)

Unstructured: https://unstructured.io/
LLama Parse: https://llamahub.ai/l/readers/llama-index-readers-llama-parse?from=readers

Context

Definition: The context size refers to the maximum number of tokens (words or subword units) that the model can process in a single input sequence. It determines how much textual information the model can consider at once when generating responses or predictions.

- A larger context size allows the model to capture longer dependencies and understand more extensive context within the input, leading to more coherent and relevant outputs.
- A smaller context size limits the amount of information the model can utilize from the input text.

Variable Sequence Length Training for Long-Context Large Language Models

How to chunk documents ?

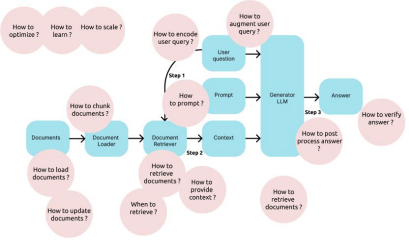
Chunking

To avoid context limitations, we can do document chunking:

- Chunk by document if the document is small
- Chunk by title or header if the document is big

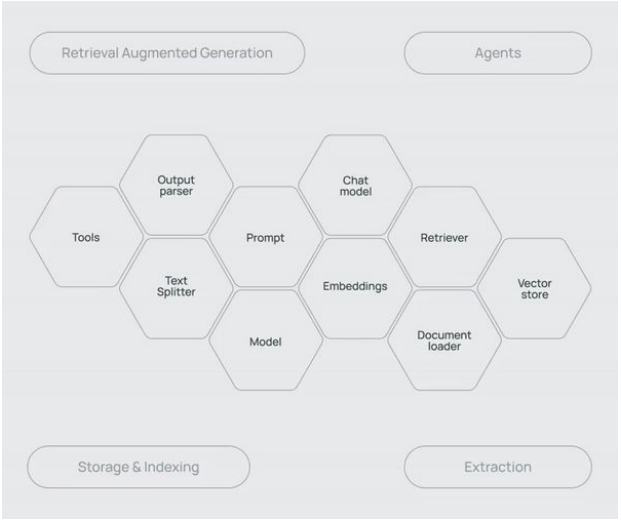
```
from langchain.text_splitter import CharacterTextSplitter
text_splitter = CharacterTextSplitter(
    separator = "\n\n",
    chunk_size = 1000,
    chunk_overlap  = 200,
    length_function = len,
    is_separator_regex = False,
)
```

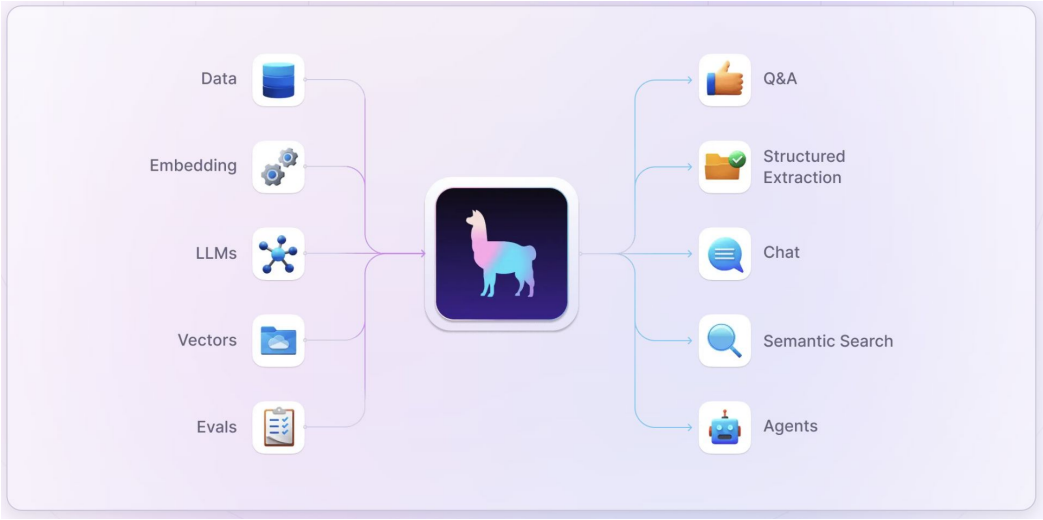
Announcing ScaNN: Efficient Vector Similarity Search



RAG Frameworks in Python

LangChain





Langchain

LlamaIndex

How to scale ?

Cloud services

Cloud Services provide:

- A Secure environment
- Enough compute to train big models
- Product as a Service (PaaS)
  - LLMs APIs
  - Vector Store management
  - Efficient Retrieval
- Monitoring tools
- …

How to verify answer ?

Retriever Evaluation: Precision & Recall @ k

Precision @k
How many **retrieved** documents are **relevant** ?

$$\text{Precision at } K = \frac{\text{Number of relevant items in } K}{\text{Total number of items in } K}$$

Recall @k
How many **relevant** documents are **retrieved** ?

$$\text{Recall at } K = \frac{\text{Number of relevant items in } K}{\text{Total number of relevant items}}$$

How to verify answer ?

Retriever Evaluation : NDCG

NDCG can take values from 0 to 1.

- NDCG equals 1 in the case of ideal ranking when items are perfectly sorted by relevance.
- NDCG equals 0 when there are no relevant objects in top-K.
- NDCG can be between 0 and 1 in all other cases. The higher the NDCG, the better.

$$\text{DCG@K} = \sum_{k=1}^{K} \frac{rel_i}{\log_2(i+1)}$$

$$\text{NDCG@K} = \frac{DCG@K}{IDCG@K}$$

How to verify answer ?

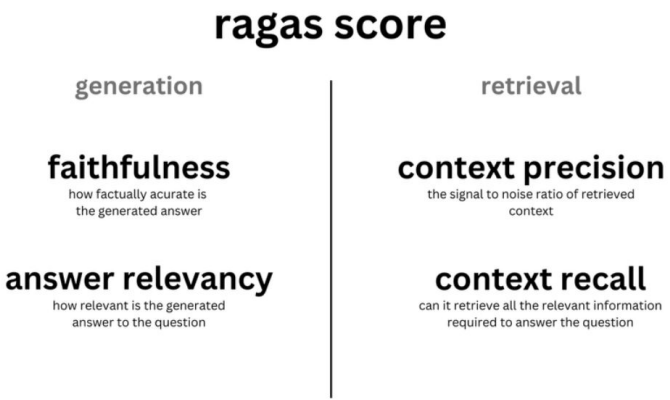
Answer Evaluation: LLM As a judge

Ask an LLM to evaluate answer quality:

- Does my answer answer to the question ?
- Does my answer used information from the context ?
- Does my answer give enough facts ?
- …

BLEU, ROUGE, Perplexity are not ideal for RAG use case.

Evaluating answers in RAG is not easy

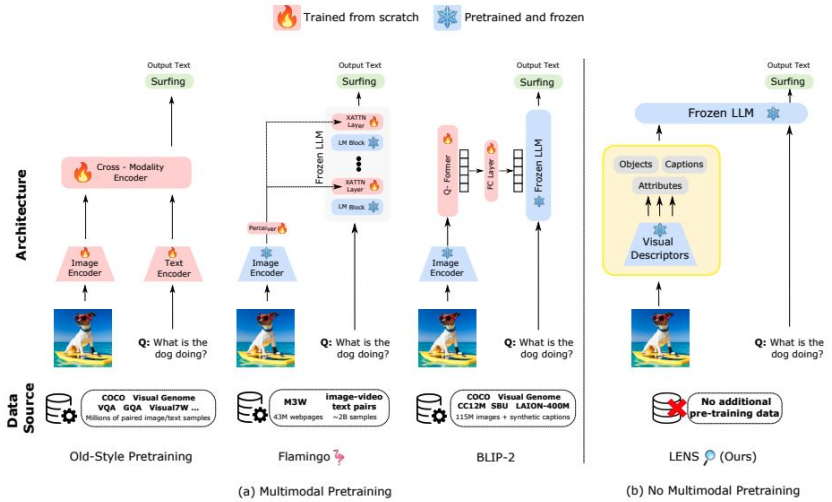


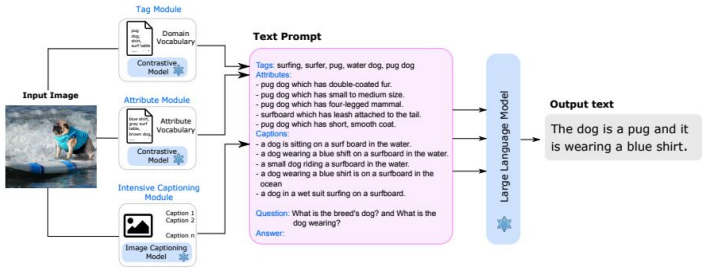
RAGAS

Multimodal

Multimodal LLM: LLM capable of processing and understanding multiple types (or “modes”) of input data, such as **text**, **images**, **audio**, **video**, and other sensory inputs, in a unified manner.

Exemple: GPT 4o, Gemini 1.5, Qwen2- VL





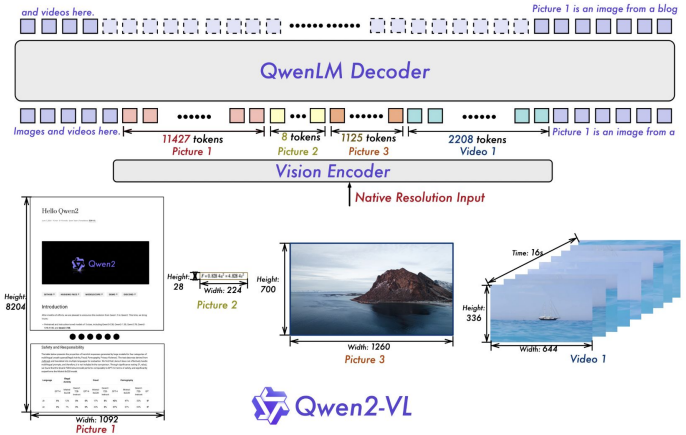
Berrios & Al, 2023, Towards Language Models That Can See: Computer Vision Through the LENS of Natural Language

Multimodal: Qwen2- VL

Outperform GPT 4o on most of the benchmarks

Multimodal Rotary Position (M-ROPE): "By deconstructing the original rotary embedding into three parts representing temporal and spatial (height and width) information, M-ROPE enables LLM to concurrently capture and integrate 1D textual, 2D visual, and 3D video positional information."

		Qwen2-VL-72B	GPT-4o-0513	Claude3.5-Sonnet	Other Best Model
College-level Problems	MMMU	64.5	**69.2**	68.3	66.1 (Gemini-2)
Mathematical Reasoning	MathVista	**70.5**	63.8	67.7	69.0 (Gemini-2)
	MATH-Vision	25.9	**30.4**	-	30.3 (GPT-4 Turbo)
Document and Diagrams Reading	DocVQA	**96.5**	92.8	95.2	94.1 (InternVL2-76B)
	ChartQA	88.3	85.7	**90.8**	88.4 (InternVL2-76B)
	OCRBench	**855**	736	788	852 (MiniCPM-V 2.6)
	MTVQA	**32.6**	27.8	25.7	23.2 (Qwen-VL-1.0-Ultra)
	InfoVQA	**84.5**	-	-	82.0 (InternVL2-76B)
	TextVQA	**85.5**	-	-	84.4 (InternVL2-76B)
General Visual Question Answering	RealWorldQA	**77.8**	75.4	60.1	72.2 (InternVL2-76B)
	MMStar	**68.3**	63.9	62.2	67.1 (InternVL2-76B)
	MMVet	**74.0**	69.1	66.0	67.5 (GPT-4o)
	MMT-Bench	**71.7**	65.5	-	63.4 (InternVL2.5)
	MMBench-1.1	**85.9**	82.2	78.5	85.5 (InternVL2-76B)
	MME	**2482.7**	2328.7	1920.0	2414.7 (InternVL2-76B)
	HallBench	**58.1**	55.0	49.9	55.2 (InternVL2-76B)
Video Understanding	MVBench	**73.6**	-	-	69.6 (InternVL2-76B)
	EgoSchema	**77.9**	72.2	-	72.2 (Gemini 1.5 Pro)
	PerceptionTest	**68.0**	-	-	66.9 (LLaVA-OneVision-72B)
	Video-MME w/o subs	71.2	71.9	60.0	**75.0** (Gemini 1.5 Pro)
	Video-MME w/ subs	77.8	77.2	-	**81.3** (Gemini 1.5 Pro)
Visual Agent	FnCall	**93.1**	90.2	-	-
	AITZ	**89.6**	70.0	-	83.0 (AITZ)
	Gym Cards	**61.7**	53.6	-	45.5 (BLAVA)
	ALFRED	**67.8**	-	-	67.7 (Task&Act)



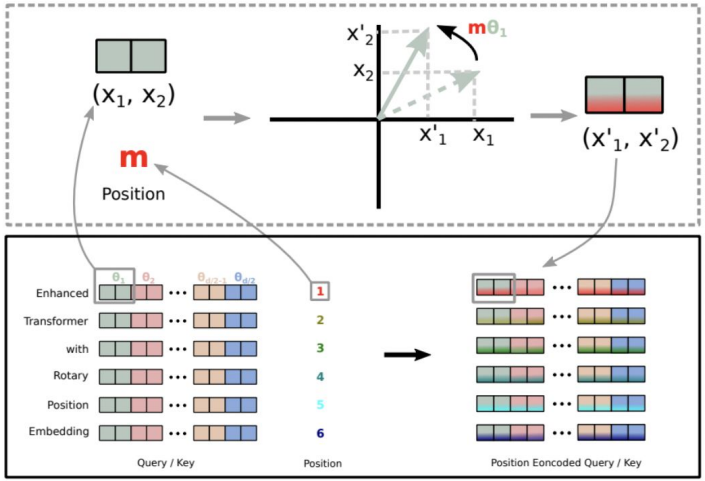
Rotary Embedding (ROPE)

Rotary Position Embedding, or RoPE, is a type of position embedding which encodes absolute positional information with rotation matrix and naturally incorporates explicit relative position dependency in self-attention formulation.

Unlike traditional position embeddings, which add fixed vectors to represent positions, RoPE encodes positional information directly into the attention mechanism by rotating the query and key vectors in the Transformer architecture. This approach allows the model to better handle long-range dependencies while maintaining the flexibility of the attention mechanism.

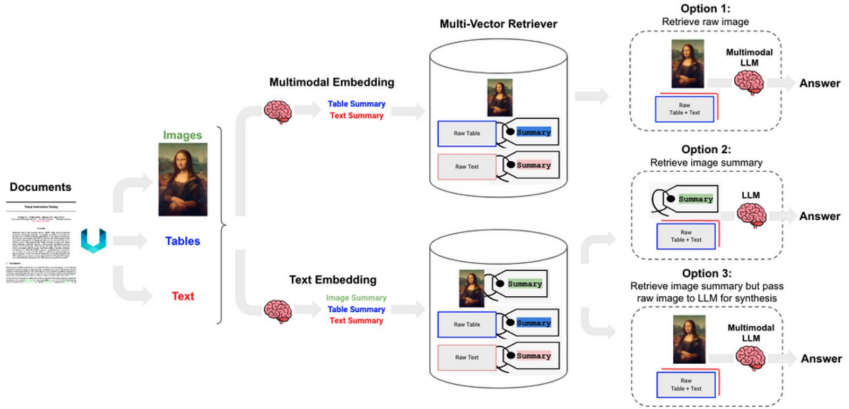
Properties:

- Flexibility of being expand to any sequence lengths
- Decaying inter-token dependency with increasing relative distances
- Capability of equipping the linear self-attention with relative position encoding.



Su & AL. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding

Multimodal RAG



Option 1: Store raw image using multimodal embedding. Retrieve images based on multimodal embedding similarity. Provide the raw image to the generator.

Option 2: Store the summary of the image using a multimodal LLM. Retrieve images based on its summary embedding. Provide the **summary** to the generator.

Option 3: Store the image and its summary using a multimodal LLM. Retrieve images based on its summary embedding. Provide the **image** to the generator.

LangChain, Multi-Vector Retriever for RAG on tables, text, and images

Multimodal RAG

Michihiro Yasunaga,[1 *] Armen Aghajanyan,[2] Weijia Shi,[3] Rich James,[2] Jure Leskovec,[1] Percy Liang [1]
Mike Lewis,[2] Luke Zettlemoyer,[3 2] Wen-tau Yih [2]

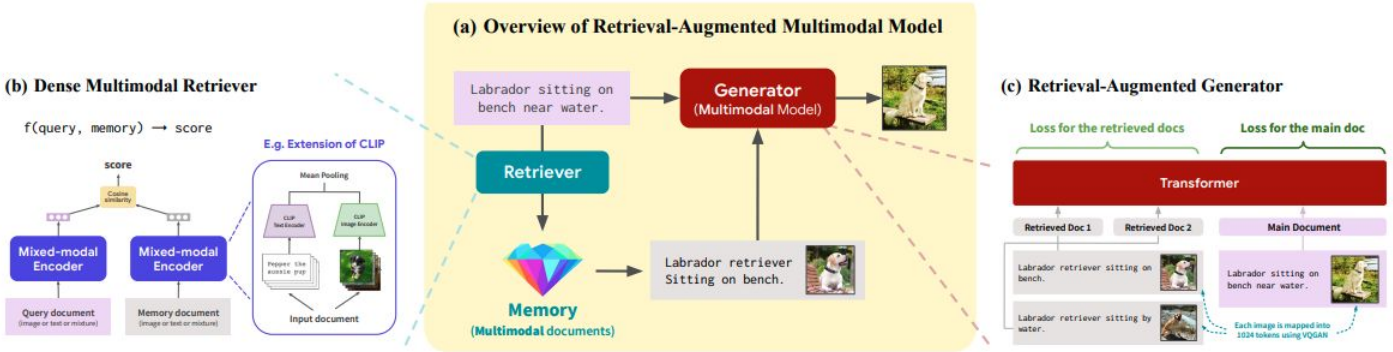
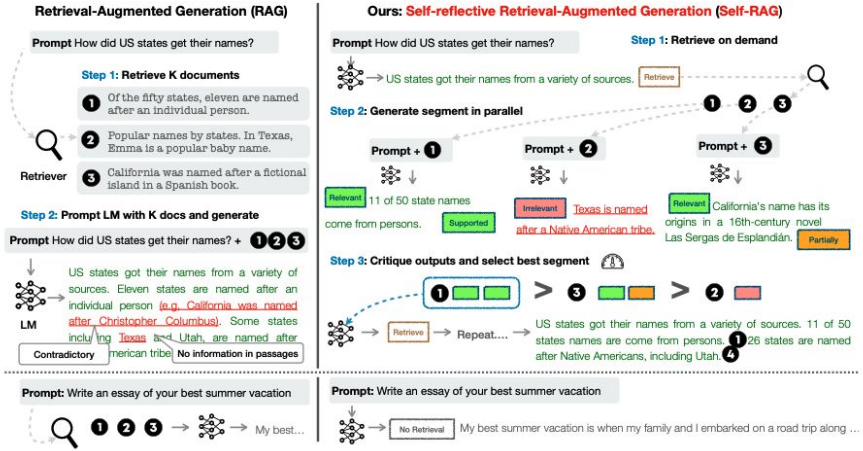


Figure 1. **Our approach, retrieval-augmented multimodal modeling**. **(a)** Overview: given an input multimodal document, we use a **retriever** to retrieve relevant multimodal documents from an external memory, and use the **generator** to refer to the retrieved documents and make predictions for the input (e.g., generate the continuation). **(b)** The multimodal retriever is a dense retriever with a mixed-modal encoder that can encode mixture of text and images (§3.2). **(c)** The retrieval-augmented generator uses the CM3 Transformer architecture, and we prepend the retrieved documents to the main input document that we feed into the model (§3.3).

Yasunaga & Al, 2023, Retrieval-Augmented Multimodal Language Modeling

Self-RAG

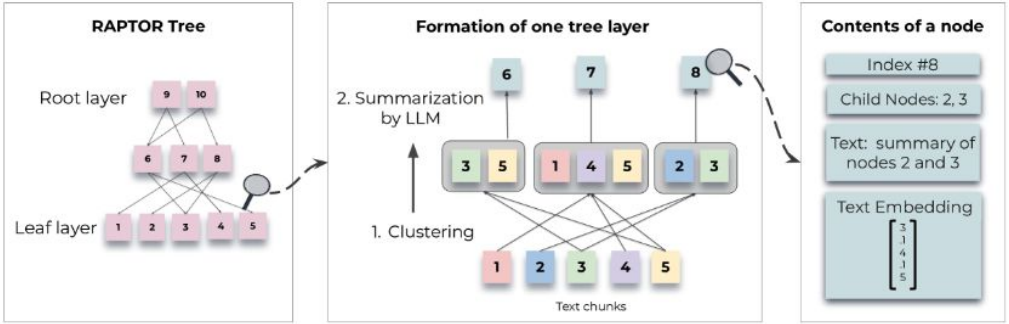
- Generates multiple possible response segments in parallel, utilizing the retrieved documents as context.
- The model ranks the generated segments based on their critique scores, selecting the most accurate and relevant segment as the final output.
- This selection process ensures that the response is both factually correct and contextually appropriate.



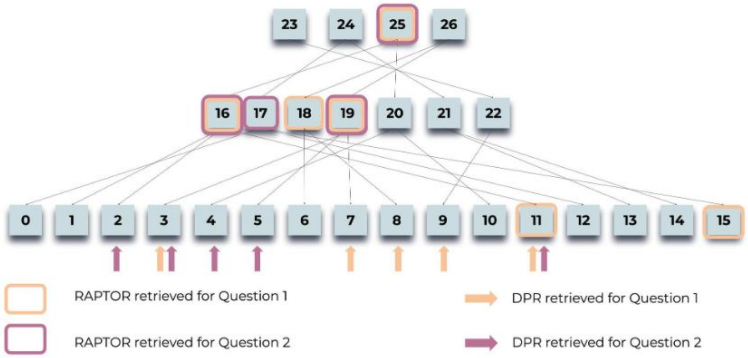
Asai & Al, 2023, Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection

Ahmed, 2024, SELF-RAG (Self-Reflective Retrieval-Augmented Generation): The Game-Changer in Factual AI Generation [Blog]

RAPTOR



- RAPTOR recursively summarizes retrieved documents. Instead of processing the full text of multiple documents directly, it creates concise summaries that retain the most important information at each recursive step.
- This hierarchy of summaries reduces the amount of information that needs to be processed while preserving the context and key facts from the original documents.

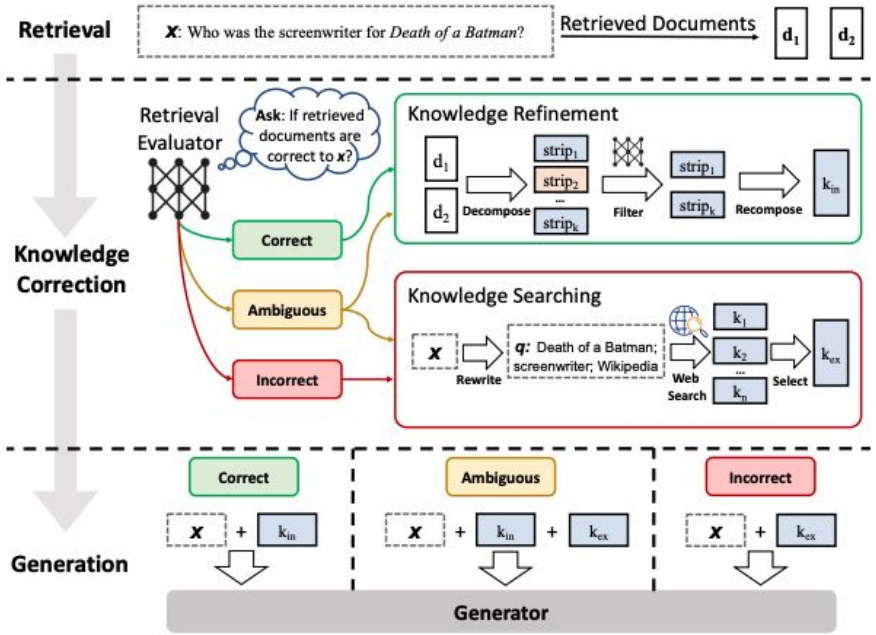


Sarthi & AL, 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval

Ahmed, 2024. SELF-RAG (Self-Reflective Retrieval-Augmented Generation): The Game-Changer in Factual AI Generation [Blog]

Corrective RAG (CRAG)

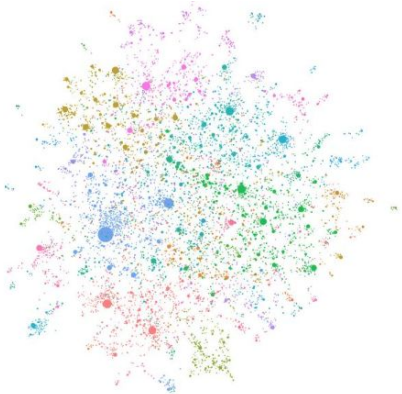
- Add a retrieval evaluator based on the quality of retrieved sources
- If sources are considered as incorrect, or ambiguous, augment or replace the context by a Web Search query



Yan & Al, 2024, Corrective Retrieval Augmented Generation

GraphRAG

- The LLM processes the entire private dataset, creating references to all entities and relationships within the source data, which are then used to create an LLM-generated knowledge graph.
- This graph is then used to create a bottom-up clustering that organizes the data hierarchically into semantic clusters. This partitioning allows for pre-summarization of semantic concepts and themes, which aids in holistic understanding of the dataset.
- At query time, both of these structures are used to provide materials for the LLM context window when answering a question.



Microsoft, 2024, GraphRAG: Unlocking LLM discovery on narrative private data

Conclusion

Simple RAG: Encodes document content into a vector store, enabling quick retrieval of relevant information to enhance model responses.
Context Enrichment: Adds surrounding context to each retrieved chunk, improving the coherence and completeness of the returned information.
Multi-faceted Filtering: Applies various filtering techniques (metadata, similarity thresholds etc.) to refine and improve the quality of retrieved results.
Fusion Retrieval: Combines vector-based similarity search with keyword-based retrieval to improve document retrieval.
Intelligent Reranking: Reassesses and reorders initially retrieved documents to ensure that the most pertinent information is prioritized for subsequent processing.
Query Transformation: Modifies or expands the original query with query rewriting, step-back prompting and sub-query decomposition.
Hierarchical Indices: First identifies relevant document sections through summaries, then drills down to specific details within those sections.
Hypothetical Questions: HyDE transforms queries into hypothetical documents that contain answers, bridging the gap between query and document distributions in vector space.
Choose Chunk Size: Selects an appropriate fixed size for text chunks to balance context preservation and retrieval efficiency.
Semantic Chunking: Unlike traditional methods that split text by fixed character/word counts, semantic chunking creates more meaningful, context-aware segments.
Context Compression: Compresses and extracts the most pertinent parts of documents in the context of a given query.
Explainable Retrieval: Not only retrieves relevant documents based on a query but also provides explanations for why each retrieved document is relevant.
Retrieval w/ Feedback: Utilizes user feedback on the relevance and quality of retrieved documents and generated responses to fine-tune retrieval and ranking models.

Conclusion

Adaptive Retrieval: Classifies queries into different categories and uses tailored retrieval strategies (factual, analytical, contextual etc.) for each, considering query context and preferences.
Iterative Retrieval: Analyzes initial results and generates follow-up queries to fill in gaps or clarify information.
Ensemble Retrieval: Applies different embedding models or retrieval algorithms and uses voting or weighting mechanisms to determine the final set of retrieved documents.
Graph RAG= Retrieves entities and their relationships from a knowledge graph relevant to the query, combining with unstructured text for more informative responses.
Multimodal: Integrates models that can retrieve and understand different data modalities, combining insights from text, images, and more.
RAPTOR: Uses abstractive summarization to recursively process and summarize retrieved documents, organizing the information in a tree structure for hierarchical context.
Self RAG: Multi-step process integrating decision, document retrieval, relevance filtering and generative feedback for more powerful model responses.
Corrective RAG: Dynamically evaluates and corrects the retrieval process, combining vector databases, feedback, and models to improve responses.
Few shot examples: Provides a few examples in the prompt to help the LLM understand the desired output